IRM Software Overview

Robert Goodwin

Wed, Nov 16, 1994

Introduction

Internet Rack Monitor software is an evolution of that used in several front end control systems at Fermilab and elsewhere. This latest version runs on the MVME162-22 cpu board with MC68040 cpu, 4MB dynamic ram, 0.5MB static ram, 1MB flash memory, ethernet interface, and support for up to four IndustryPack daughter boards. The latter allows connection to I/O signals via ribbon cables to digital and analog interface boards mounted inside the IRM chassis. The ethernet interface allows network connection and supports widely-used Internet protocols that allow data request and setting access as well as alarm reporting, all based upon the UDP (User Datagram Protocol) transport layer.

Local database

The nonvolatile memory houses a number of configuration tables that characterize each station's installation. Included in these tables is a local database for analog channels and digital bits. It includes text and scale factors used by local control applications for scaling as well as alarms reporting. It also houses down loaded code for local and page applications in a memory-resident file system. Local applications are used for closed loop support and for system extensions such as TFTP protocol server support. Page applications support a virtual console access to the system for local control, configuration and diagnostics use. The system code is acquired from a server station by the prom-based 162bug via the TFTP protocol at boot time.

Cyclic data pool activities

IRM software is a collection of tasks that use the pSOS operating system kernel. The primary activities of the system are synchronized by an external timing signal, which may be an external trigger input, or it may be decoded from a "Tevatron clock" signal. (In the absence of a synchronizing signal at 10Hz or 15Hz, the system operates asynchronously at 12.5Hz.) At the beginning of each cycle, the data pool is refreshed according to instructions in the nonvolatile Data Access Table that are interpreted at the start of each cycle. Also, all active local applications are run to operate on the fresh data for closed loop jobs or more complex data pool updates. (A local application is compiled and downloaded separately from the system code itself.) After the data pool is refreshed, all active data requests having replies that are due on the present cycle are fulfilled and delivered to network requesters. Alarm scanning is performed on all analog channels and binary status bits that are enabled for such monitoring. Finally, the currently active page application is run. So, the data pool is accessed by local application activities, replies to data requests, alarm scanning, and the current page application.

Synchronization

The Tevatron clock signal can carry up to 256 events. The IndustryPack digital board decodes each event and interrupts the cpu, allowing time-stamping of each event. Status bits derived from this event activity are part of the data pool and can be used to synchronize data pool updating and local application

data requests for up-to-1000Hz data available from the IndustryPack analog board; i.e., one can plot 300Hz data, say, with time stamps measured from a selected clock event.

Data request protocols

Due to the evolution of this front end system, three different data request protocols are supported. The original request protocol is called Classic; it is used by stations that communicate among themselves as well as by the Macintosh-based parameter page application developed by Bob Peters. A second protocol was designed by the Fermilab D0 detector people to fulfill their specific needs. The third is that used by the Acnet control system at Fermilab.

Data server logic is included for both Classic and Acnet request protocols. This allows one station (the server) to be targeted by a requesting host with a request for data from other stations, in which the server station forwards the request via multicasting to the other stations and compiles their individual responses into the single reply it delivers to the requesting host. The purpose of this logic is to reduce the number of replies a host might have to endure, in response to a request for data from many different stations. The server station can do this efficiently because of the built-in logic in each station that combines multiple replies—due on the same cycle to the same destination—into a single network datagram. For example, suppose three hosts make requests for data from the same 10 stations at 10Hz, and each host uses the same server node. The server node will receive 10 composite replies each 10Hz cycle, and it will send 3 replies to the three hosts, for a total of 130 frames/second, thus requiring each host to receive only 10 frames/sec. Without the server station, each host would have to receive 100 frames/sec.

Alarm handling

Alarm scanning is performed on all selected analog channels and digital bits each operating cycle, as mentioned above. When a change in alarm state (good-bad or bad-good) is detected, an alarm message is queued to the network to share this news with the outside world. Such alarm messages can be multicast, so that multiple interested hosts can learn of them. A local application can also sample such alarm messages and reformat them to target a designated host alarm server, as is required in the Acnet system. As a local diagnostic, such alarm messages can be encoded for display or printout via the station's serial port, including both locally-generated ones as well as those received by that station listening to the alarm multicast address.

Diagnostics

Several diagnostic features are included in the IRM design. The digital IndustryPack board provides test signals that are driven by interrupt and task activities. The interrupt signals are also displayed on 8 LEDs. These signals can be connected to a logic analyzer to capture timing and related program activities of the station's operation.

A suite of page applications is available to perform various diagnostic displays via the virtual console support.

blocks of memory, or copy them from one station to another.

2: Display Tevatron clock events with 10Hz updating. Capture and display network frame activity, providing a kind of built-in "poor man's sniffer", in which the timing of frame reception or transmission processing is shown to 1 ms resolution within the time-of-day-specified operating cycle.

3: A network client page allows exercising the standard IP ping and UDP echo tests, as well as Network Time Protocol and Domain Name Service queries.

4: A list of kernel-allocated blocks of dynamic memory in the local station can be displayed and updated continuously.

5: All three data request protocols can be exercised in a test mode to verify particular cases and measure response times.

6: A station survey page allows listing several characteristics of a list of stations, including system code version, amount of free dynamic memory, time since last reset, number of allocated channels and bits, number of active data requests, and operating cycle time.

7: A log of recent settings performed in a given station can be displayed in a similar fashion to that of recent network frame activity.

Software development

Program preparation is done on a Macintosh using the MPW (Macintosh Programmer's Workshop) development system. The system code and application codes are both developed under MPW. The system code provides a core level of support that can be enhanced by the addition of local applications (LAs) to support the specific needs of a given installation. Such LAs may be written by a user in C or Pascal and, using the MPW tools, compiled and linked with the help of a small library of glue routines that invoke system code services. The resulting linker output is downloaded to any local station using the TFTP client also running as an MPW tool. The code received by the station's TFTP server is copied into the non-volatile memory resident file system. Using a page application for the purpose, a set of parameters are specified to be passed to the LA upon each invocation. These parameters are constant data that are often Channel#s or Bit#s, whose present readings determine the course of action of the LA. The first parameter is always an enable Bit# for the LA. When the enable bit is set, the system finds the code for the LA in the resident file system and copies it into allocated dynamic memory for subsequent execution. Every operating cycle, during Data Access Table processing, each enabled LA is called with the set of specified parameters. It can perform any short-term activity it needs to according to its context. The first time it is called, it allocates memory to maintain its own execution context. A pointer to this context memory is also passed to the LA each time it is called. An LA that is written to be a network server can also be invoked upon reception of a network message intended for it. If the enable bit is turned off, thus disabling the LA, a final call is made to the LA to allow it to release any resources. An LA can be edited, compiled, linked and downloaded to a target station without requiring a reset of the station. In the case that such an LA is already enabled and active at the time of downloading, the switch to the new version is automatic. Minor program changes can thus be accomplished in seconds.